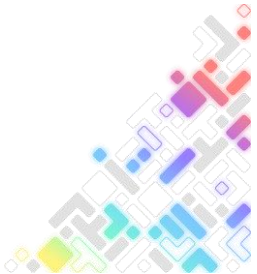
EMBEDDED OPEN SOURCE SUMMIT

Camera/ISP Drivers Using V4L2 Media Controller Framework

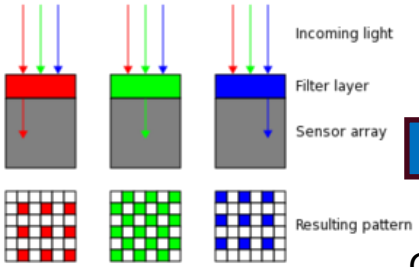
Karthik Poduval, Amazon Lab126

Evolution of Cameras

- External ISPs (camera modules) early 2000s
  - USB Cameras (UVC Protocol)
  - Parallel Interface Cameras
  - CSI & CSI2 based Cameras
- SoC’s started getting ISP
  - OMAP3
  - QUALCOMM 8xxx series SoCs



External ISP



Camera Capture Interface

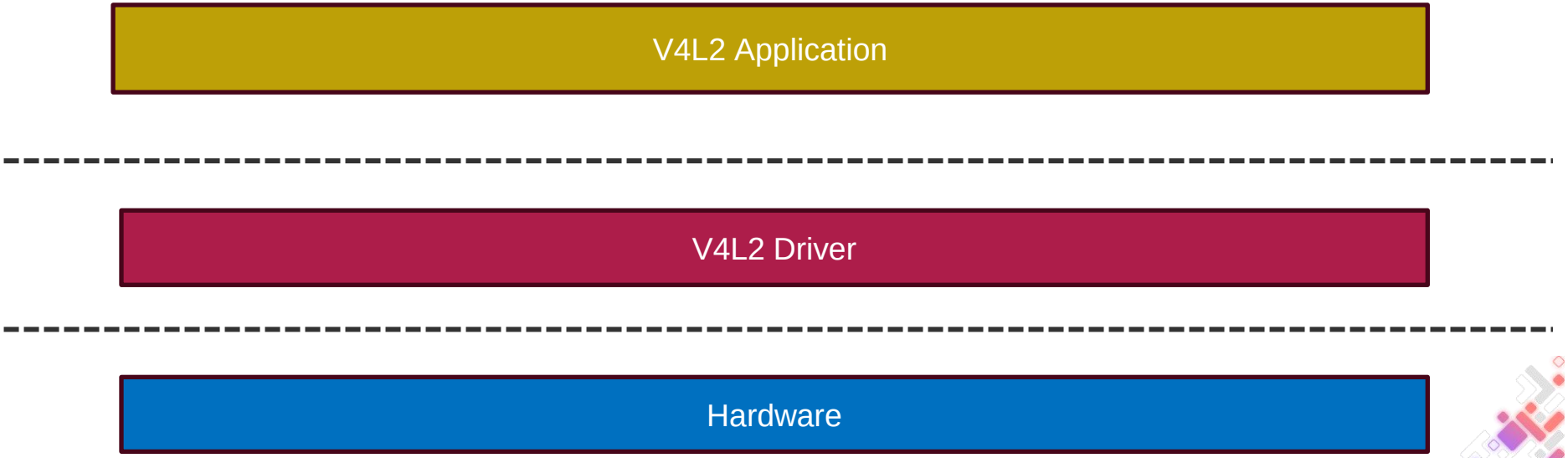
ISP

Camera Capture Interface

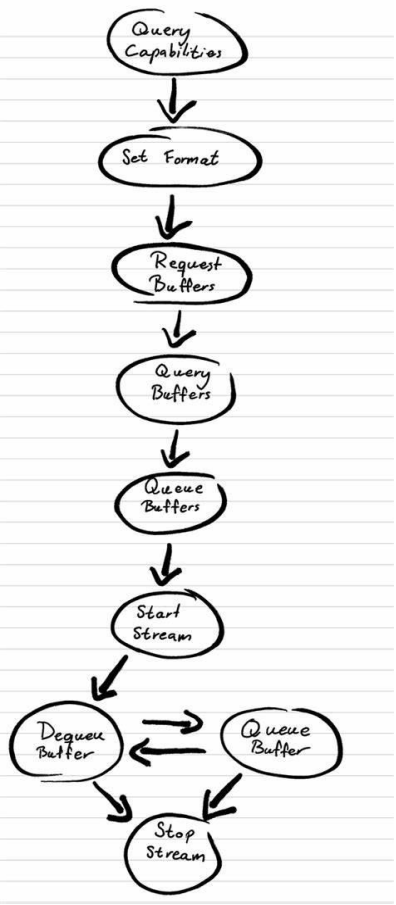
SoC / Host PC

API Abstraction Layer

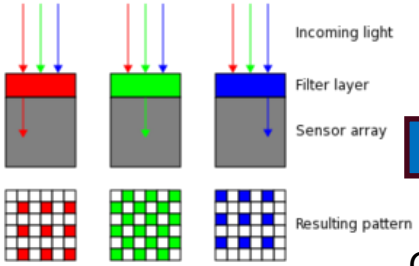
V4L2 Camera



V4L2 Application Flow



Internal ISP



Camera Capture Interface

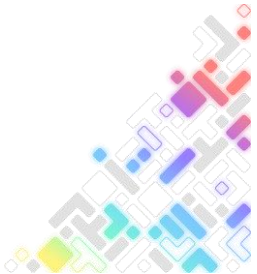
API Abstraction Layer

ISP

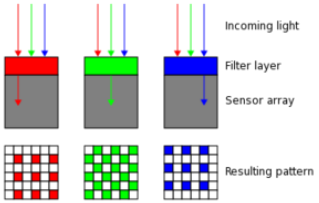
SoC / Host PC

Evolution of V4L2

- External ISPs
  - V4L2 was pretty good for this
  - Provided APIs to View Camera as a whole
    - Collection of various controls
    - An image/video stream
- Internal ISPs
  - Camera Bayer sensor placed externally
  - ISPs now contained inside the SoC
  - Complexity grown many fold as ISPs are fairly complex
  - Media controller framework augments V4L2 to help deal with such complexities



Generic ISP



I2C

CSI2

I2C Controller

CSI2/D-PHY Controller

Black Level Correction

Defective Pixel Correction

DMA

Statistics

Demosiac

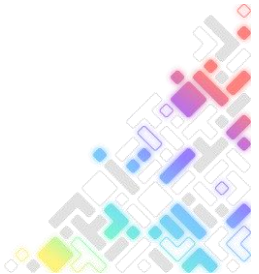
Color Correction

Scaling/Color Space Correction

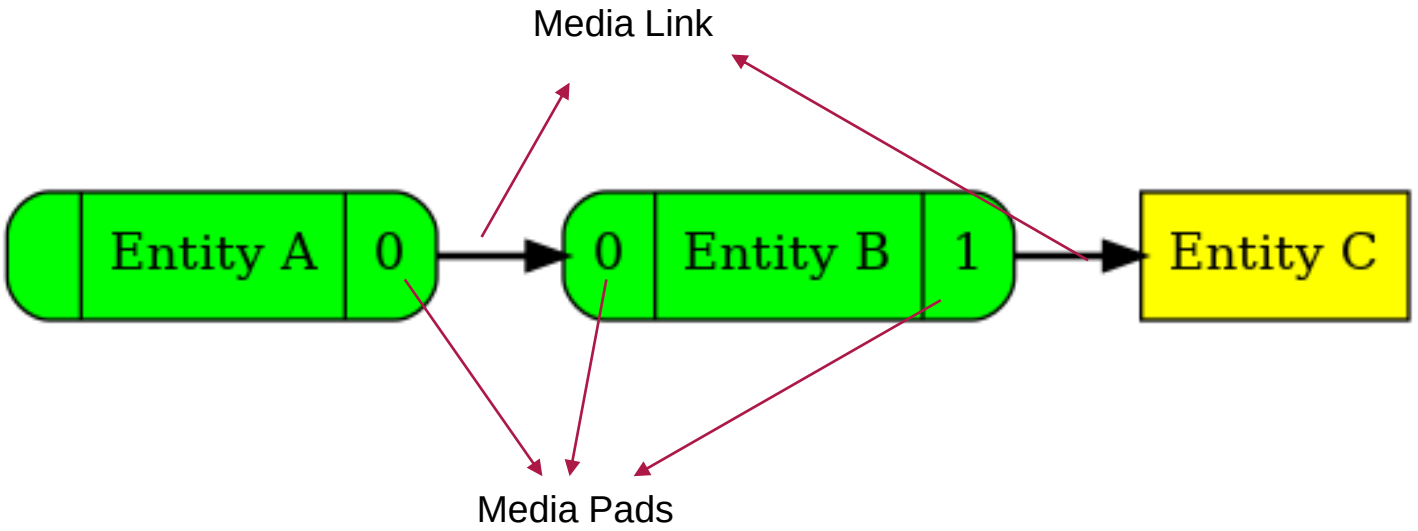
DMA

Media Controller

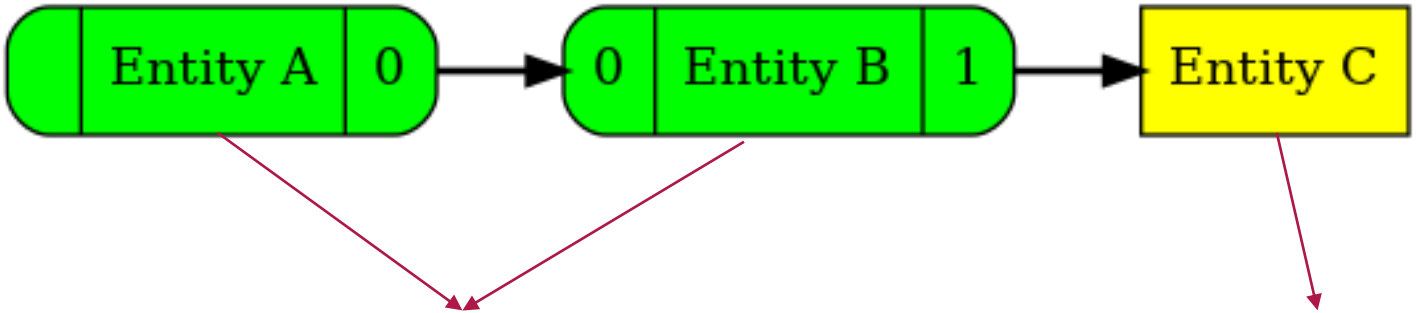
- ISPs have multiple processing blocks
  - The blocks can be connected to form processing pipelines or graphs
  - Media controller allows to represent such processing blocks in software using the media device model
- Media Device Model
  - **Media Entity:** A basic media hardware or software processing block
  - **Media Pad:** A connection pad that allows connecting one entity to another via links
  - **Media Link:** A connection between pads. These links could further be.
    - Interface Link: Between Linux kernel interface and an entity
    - Data Link: Between 2 pads in an entity
    - Ancillary Link: A logical link between two entities (ex: Lens Actuator, Flash)



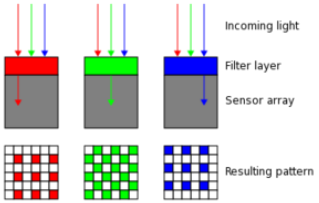
Media Device Model



Media Device & V4L2



Generic ISP



I2C

CSI2

I2C Controller

CSI2/D-PHY Controller

Black Level Correction

Defective Pixel Correction

DMA

Statistics

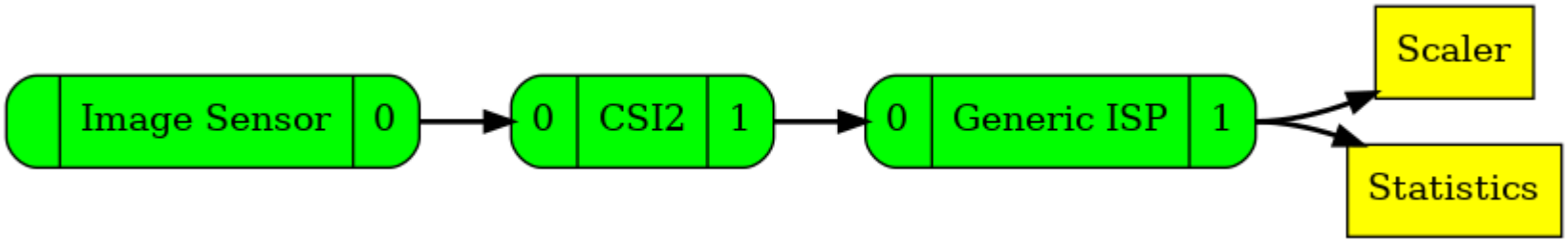
Demosiac

Color Correction

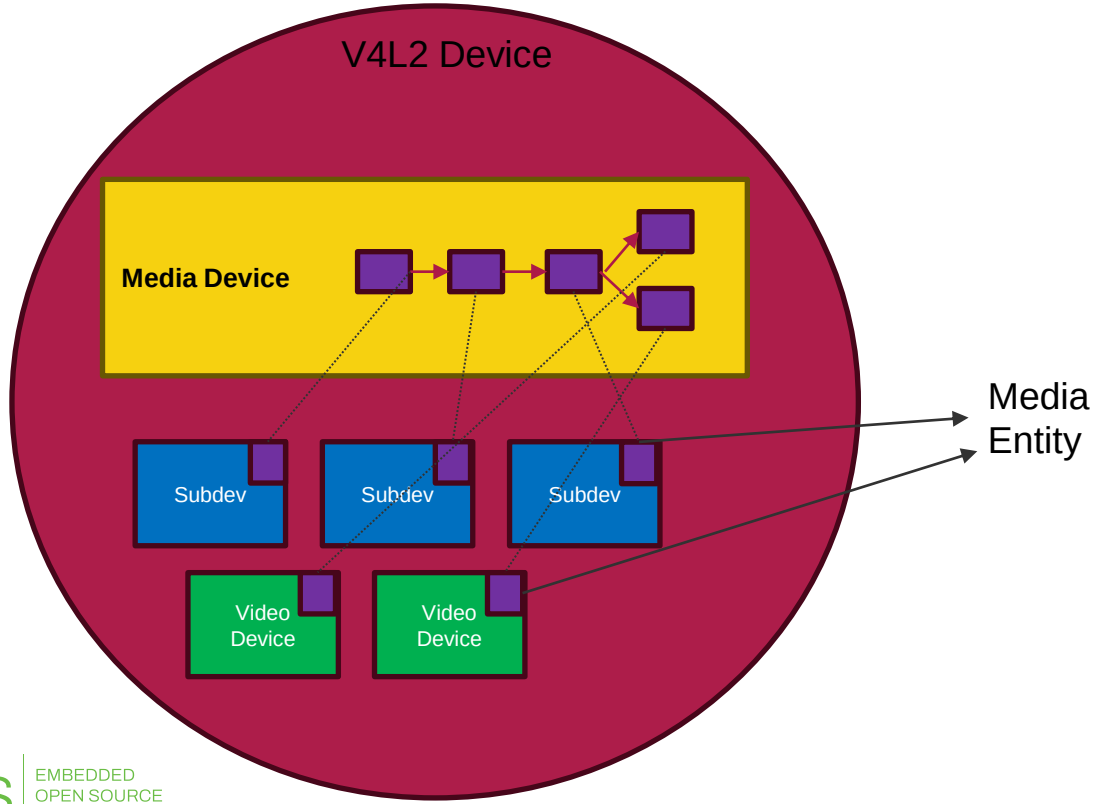
Scaling/Color Space Correction

DMA

Media Device Representation of Generic ISP



Relationship between media controller and V4L2



Media Controller API

Registration: media_device_register

- Followed by a call to kernel API media_device_init
- Created media device node ex: /dev/mediaN
- Sets file operations for the media device object

- Userspace should see the /dev/mediaN type device
- Not much to see in device node until entities register further
- Supports IOCTLs to
  - Get media device information
  - Enumerate Entities
  - Enumerate Links
  - Setup Links (connect/disconnect pads)
  - Get the Existing Graph Topology
  - Allocate a Media Request

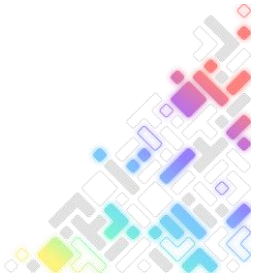
Subdevice

- Here we understand all the various subdevice Media Entity related initializations performed
- Set the media entity object's function
  - Video Composer
  - Pixel Formatter
  - Pixel Encoder/Converter
  - LUT
  - Scaler
  - Statistics
  - Encoder
  - Decoder
  - ISP
- Set the pads for the entity number of pads and initialize flags for the pads
  - Whether source or sink
  - Indicate if pad must connect and can't be left dangling
- Set the Media Entity Ops
  - Get FWNODE pad: Used to determine pad number for async graph devices
  - Link Setup: Notify entity of link changes
  - Link Validate: Function that checks the entity pads connected via link are compatible i.e. sink and src have same format, size etc.

- Userspace can scan the media pipeline find subdev entities discover
  - Number of pads and whether source or sink
  - The function of the entity

Video Device

- Just like subdev call media_entity_pads_init to setup pad, typically one for a video device
- Also have media entity ops just like the case with subdevs



Other helper API

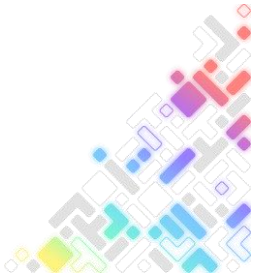
- media_entity_setup_link: to connect two Media Pads
- media_entity_find_link: to find the link between two pads
- media_entity_remote_pad_unique, media_pad_remote_pad_first: to find pads on async graph subdevs
- media_entity_to_v4l2_subdev: To get subdev from media entity
- media_create_ancillary_link: For Lens and Auto Focus Actuator Subdevices

- Userspace can get media graph topology by calling MEDIA_IOC_G_TOPOLOGY
  - Used by media-ctl –p
- Userspace can setup links using ioctl MEDIA_IOC_SETUP_LINK
  - Used by media-ctl –links

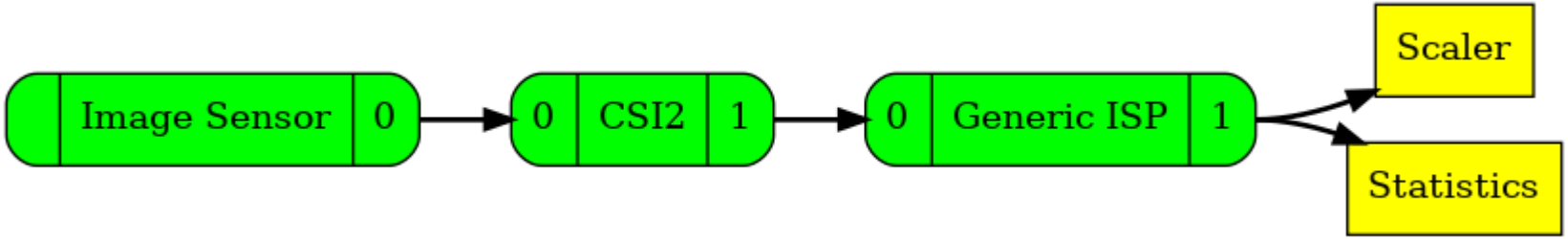
Pipeline API

- media_pipeline_start: Starts the media pipeline and increments a start_count member
- media_pipeline_stop: Stops the media pipeline
- media_graph_walk_init: Initializes a graph walk
- media_graph_walk_start: Start the graph walk
- media_graph_walk_next: Used to get next entity in media graph
- media_graph_walk_cleanup: Cleanup after the graph walk

- These APIs are typically called from drivers to handle complex ISP pipelines



Example



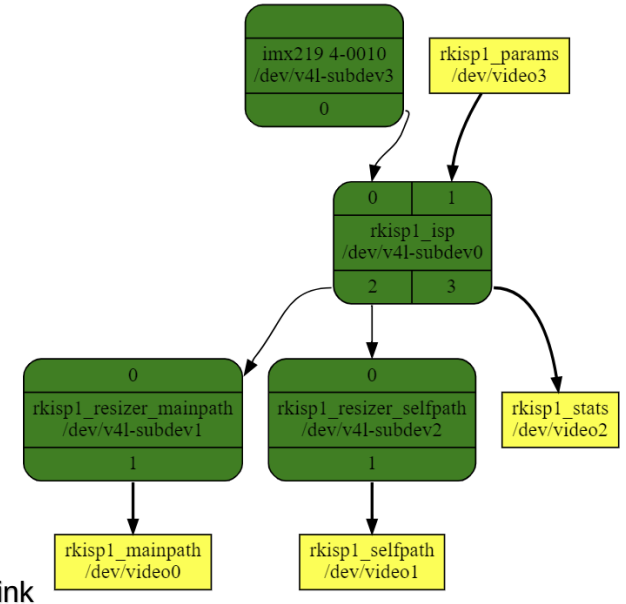
- Image Sensor and CSI2 may be different drivers from ISP driver that includes ISP, Scaler and Statistics
- V4L2 Async framework is used to find out (via device tree) how they are connected to the media pipeline and using a Async Notifier's bound and complete (media device owner only) callback, the links can be made on runtime ex: Bind correct Image Sensor based on the HW board version via device tree
- Once all entities are registered, a Stream ON on one of the Video Devices (Yellow Boxes) will also turn on the pipeline. ISP driver can do graph walk API to find all the associated subdevices and stream them on. It can also keep track of start_count to avoid turning on the pipeline twice

media-ctl example

media-ctl -d "platform:rkisp1" -p

media-ctl -d "platform:rkisp1" --print-dot

```
Media device information
------------------------
driver          rkisp1
model           rkisp1
serial
bus info
hw revision     0x0
driver version  0.0.0

Device topology
- entity 1: rkisp1-isp-subdev (4 pads, 5 links)
            type V4L2 subdev subtype Unknown flags 0
            device node name /dev/v4l-subdev0
        pad0: Sink
                [fmt:SBGGR10_1X10/800x600 field:none
                 crop.bounds:(0,0)/800x600
                 crop:(0,0)/800x600]
                <- "rockchip-sy-mipi-dphy":1 [ENABLED]
        pad1: Sink
                <- "rkisp1-input-params":0 [ENABLED]
        pad2: Source
                [fmt:SBGGR10_1X10/800x600 field:none
                 crop.bounds:(0,0)/800x600
                 crop:(0,0)/800x600]
                -> "rkisp1_selfpath":0 [ENABLED]
                -> "rkisp1_mainpath":0 [ENABLED]
        pad3: Source
                -> "rkisp1-statistics":0 [ENABLED]

- entity 2: rkisp1_selfpath (1 pad, 1 link)
            type Node subtype V4L flags 0
            device node name /dev/video0
        pad0: Sink
                <- "rkisp1-isp-subdev":2 [ENABLED]

- entity 3: rkisp1_mainpath (1 pad, 1 link)
            type Node subtype V4L flags 0
            device node name /dev/video1
        pad0: Sink
                <- "rkisp1-isp-subdev":2 [ENABLED]

- entity 4: rkisp1-statistics (1 pad, 1 link)
            type Node subtype V4L flags 0
            device node name /dev/video2
        pad0: Sink
                <- "rkisp1-isp-subdev":3 [ENABLED]

- entity 5: rkisp1-input-params (1 pad, 1 link)
            type Node subtype V4L flags 0
            device node name /dev/video3
        pad0: Source
                -> "rkisp1-isp-subdev":1 [ENABLED]

- entity 6: rockchip-sy-mipi-dphy (2 pads, 2 links)
            type V4L2 subdev subtype Unknown flags 0
            device node name /dev/v4l-subdev1
        pad0: Sink
                [fmt:SRGGB10_1X10/3280x2464 field:none]
                <- "imx219 2-0010":0 [ENABLED]
        pad1: Source
                [fmt:SRGGB10_1X10/3280x2464 field:none]
                -> "rkisp1-isp-subdev":0 [ENABLED]

- entity 7: imx219 2-0010 (1 pad, 1 link)
            type V4L2 subdev subtype Sensor flags 0
            device node name /dev/v4l-subdev2
        pad0: Source
                [fmt:SRGGB10_1X10/3280x2464 field:none]
                -> "rockchip-sy-mipi-dphy":0 [ENABLED]
```



Setup Link

"media-ctl" "-d" "platform:rkisp1" "-l" "'imx219 4-0010':0 -> 'rkisp1_isp':0 [1]"

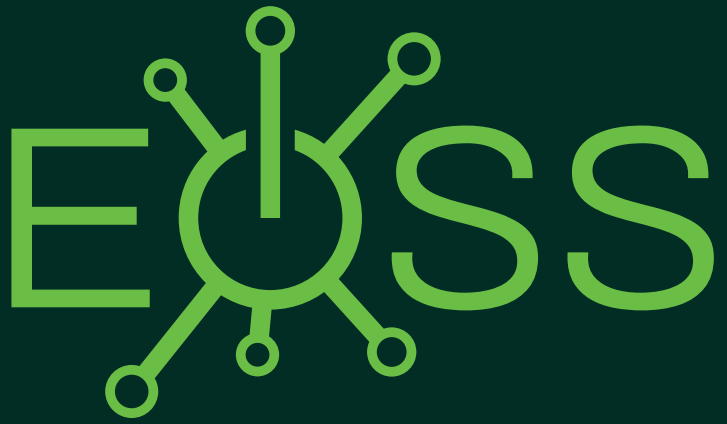
Media Request API ?

- Needs a separate session of its own
  - something for next ELC perhaps

References

- Media controller (core and V4L2) [LWN.net]
- Capture a picture with V4L2 - Marcus Folkesson
- v4l2-workflow.png (552×1349) (marcusfolkesson.se)
- Part IV - Media Controller API — The Linux Kernel documentation
- 7.17. Rockchip Image Signal Processor (rkisp1) — The Linux Kernel documentation

Questions ?



EMBEDDED
OPEN SOURCE
SUMMIT